Artificial Intelligence Algorithms for Power Allocation in High Throughput Satellites: A Comparison

Juan Jose Garau Luis, Nils Pachler, Markus Guerster, Inigo del Portillo, Edward Crawley, Bruce Cameron
Massasachusetts Institute of Technology
77 Massachusetts Av. 33-409
Cambridge, MA 02139
{garau, pachler, guerster, portillo, crawley, bcameron}@mit.edu

Abstract—**Automating resource management strategies is a key priority in the satellite communications industry. The future landscape of the market will be changed by a substantial increase of data demand and the introduction of highly flexible communications payloads able to operate and reconfigure hundreds or even thousands of beams in orbit. This increase in dimensionality and complexity puts the spotlight on Artificial Intelligence-based dynamic algorithms to optimally make resource allocation decisions, as opposed to previous fixed policies. Although multiple approaches have been proposed in the recent years, most of the analyses have been conducted under assumptions that do not entirely reflect operation scenarios. Furthermore, little work has been done in thoroughly comparing the performance of different algorithms.**

In this paper we compare some of the recently proposed dynamic resource allocation algorithms under realistic operational assumptions, addressing a specific problem in which power needs to be assigned to each beam in a multibeam High Throughput Satellite (HTS). We focus on Genetic Algorithms, Simulated Annealing, Particle Swarm Optimization, Deep Reinforcement Learning, and hybrid approaches. Our multibeam operation scenario uses demand data provided by a satellite operator, a full radio-frequency chain model, and a set of hardware and time constraints present during the operation of a HTS. We compare these algorithms focusing on the following characteristics: time convergence, continuous operability, scalability, and robustness. We evaluate the performance of the algorithms against different test cases and make recommendations on the approaches that are likely to work better in each context.

TABLE OF CONTENTS

1. INTRODUCTION

Motivation

In the coming years, the competitiveness in the satellite communications market will be largely driven by the operators' ability to automate part of their systems' key processes, such as capacity management or telemetry analysis. Companies will rely on autonomous engines to make decisions over their operation policies in order to adapt to faster and larger changes in their customer pools, and to have a better management of their systems efficiency [1].

Two tendencies settle this new scenario: a shift from static communications payloads to highly-flexible payloads [2] and an increasing demand for data [3]. The former responds to the recent improvements in multibeam satellite technology, where the number of beams and individually-configurable parameters in orbit is growing exponentially – the power, bandwidth, pointing direction, and sizing of each beam will be individually tunable. The latter tendency includes the growing necessity for data transmission through satellite links, since services such as guaranteeing connectivity in isolated regions or providing streaming capabilities in planes and ships are becoming more frequent.

In the specific case of Dynamic Resource Management (DRM) in multibeam High Throughput Satellites (HTS), while traditional approaches were based on static, human-controlled policies that relied on conservative operational margins, new systems will include dynamic algorithms capable of handling the increasing dimensionality and the rapidly-changing nature of the demand [4]. Consequently, these systems will be prepared to make quick decisions on multiple parameters per beam across thousands of active spot beams. These decisions will be taken simultaneously and repeatedly, responding to every change in the constellation's environment.

Multiple algorithms have already been proposed as possible alternatives to current human decision-making processes. Specifically, in the recent years, the spotlight has been placed on Artificial Intelligence (AI) algorithms given its success in other areas [5]. These algorithms range from population-centered approaches to Machine Learning-based methods. While one can easily find studies emphasizing the potential of each individual approach, there has been little effort to characterize and compare these methods under the same premises. In order to study their suitability for the upcoming resource allocation challenges, in this paper we try to close this research-gap by comparing the most recent and popular AI algorithms in solving the dynamic power allocation problem for multibeam satellites. We test and contrast each method under equal and realistic operational assumptions, using the same full-chain RF model and a varied range of datasets as a support.

Literature Review

The resource allocation problem for multibeam communication satellites is a well-studied NP-hard [6] and non-convex [7] problem. Furthermore, the number of available optimization variables is increasing and hence the dimensionality of the problem adds a new layer of complexity. As a result, many classic optimization methods require relaxations or large

978-1-7281-2734-7/20/$31.00 ©2020 IEEE

amounts of computing resources, leading to an inefficient performance during real-time operations.

AI is a potential solution to overcome these issues. Metaheuristic algorithms [8] are a popular group of AI algorithms that have been thoroughly studied in the context of the DRM problem. In [6] a Genetic Algorithm (GA) is used to dynamically allocate power in a 37-beam satellite and is compared to other metaheuristic approaches. A similar GA formulation is extended to include bandwidth optimization in [9], which demonstrates the advantages of dynamic allocation based on two 37-beam and 61-beam (Viasat-1) scenarios. Joint power and carrier allocation is also proposed in [10], where a two-step heuristic optimization is carried out iteratively for a 84-beam use case.

Other authors have opted for approaches based on the Simulated Annealing (SA) algorithm. In [7] a single-objective and discrete power and bandwidth allocation problem was formulated using SA and applied to a 200-beam case, with a focus on fairness. A prior hybrid optimization stage combining the GA and the SA was also considered in [6]. Fairness is a relevant topic that has also been considered in [11], where an iterative dual algorithm is proposed to optimally and fairly allocate power in a 4-beam setting.

Besides population-based and annealing approaches, swarm-based methods have also been applied to the DRM problem. In [12] the Particle Swarm Optimization (PSO) algorithm is applied to a 16-beam satellite in order to optimally allocate power.

While the performance of the presented methods has been compared to equal-allocation or heuristic-based approaches, authors have only considered the *offline* performance when presenting their results. None of these studies shows the results of continuously applying their respective methods in a dynamic, time-changing environment. Therefore, the adequacy of these algorithms to *online* scenarios has yet to be tested and contrasted.

As a potential alternative for repeated uses of the optimization tool, some studies have recently focused on Machine Learning algorithms, specifically Deep Reinforcement Learning (DRL) architectures, which address the needs of fast online performances. In [13], authors use a DRL architecture to control a communication channel in real time considering multiple objectives. Alternatively, DRL has also been exposed to multibeam scenarios, as in [14], where a Deep Q network was used to carry out channel allocation in a 37-beam scenario. Then, in [15] a continuous DRL architecture is used to allocate power in a 30-beam HTS, showing a 1,300-times speed increase with respect to a comparable GA approach. Finally, in [16] DRL is used to carry out beam-hopping tasks in a 10-beam and 37-cell scenario, showing a stable performance throughout a 24-hour test case.

The cited works show that DRL architectures have been shown to reach good solutions and be capable of exploiting the time and spatial dependencies of the DRM problem. However, most of the test cases focus on optimality and leave other important features such as robustness out of the studies. A lack of robustness in a DRL architecture might lead to non-desirable allocations in cases in which the input does not match the average behaviour of the environment. The fact that these algorithms must go through a prior training stage raises the question of how a DRL training process should be designed in order to result in robust architectures ready to be operable.

Research Goals

Apart from the lack of comparative analyses in most of the studies presented, we also identify two important modeling decisions that do not entirely reflect the future of satellite communications: first, while the majority of papers highlight the dynamic nature of the algorithms, there are no results on their performance on a continuous execution; in which repeated use of the algorithms is necessary and computing time is a limiting factor. Second, none of the studies explores the scalability of its approach. The expected dimensionality of the problem lies in the range of hundreds to thousands of beams, whereas the performance of the algorithms has mostly been assessed for scenarios with less than 100 beams.

In this paper we consider and implement the four AI algorithms introduced in the literature review – GA, SA, PSO, and DRL – alongside two hybrid approaches – SA-GA and PSO-GA – and analyze their performance on the same tasks under a realistic operational context. The main objectives addressed by this paper are:

1. To provide a formulation of the most well-studied AI algorithms in literature for solving the dynamic power allocation problem.
2. To compare and contrast the performance of these algorithms under the same models in a set of different test scenarios.
3. To characterize the online performance of the algorithms given computing time restrictions.
4. To provide scalability and robustness results for the algorithms that show the best performance in the reference cases.

Our objective is to offer the reader a comprehensive and fair comparison of these methods in order to guide the algorithm downselection process for future DRM or similar problems.

Paper Structure

The remainder of this paper is structured as follows: Section 2 describes the problem statement and the formulation used throughout the rest of this work; Section 3 presents the implementations for each of the algorithms studied; Section 4 covers the satellite, link budget, and demand models considered; Section 5 discusses the performance of the algorithms in multiple scenarios; and finally Section 6 outlines the conclusions of the paper.

2. Problem Statement

This section covers, first, a detailed problem formulation with the assumptions considered; and second, an overview of the different objective functions that will be used throughout the rest of the paper.

Problem formulation

In this study we consider a multibeam High Throughput Satellite (HTS) with N_b non-steerable beams. These beams are already pointed to a location with a defined shape. A sequence of timesteps $\{1, ..., T\}$ represents all instants in which the satellite is requested a certain throughput demand per beam. The goal of the problem is, at every timestep, to allocate a sufficient amount of power to each beam to satisfy the demand and constraints imposed by the system while minimizing resource consumption.

At a given timestep t, the demand requested at beam b is represented by $D_{b,t}$. Likewise, the power allocated to beam

b at timestep t and the data rate attained when doing so are denoted as $P_{b,t}$ and $R_{b,t}$, respectively. There is an explicit dependency between the data rate achieved and the power allocated to a particular beam, as will be covered in Section 4.

We assume the satellite, at any moment, has a total available power of P_{tot}. Similarly, every beam b has a maximum power constraint, denoted by P_b^{max}. Apart from maximum total and individual beam power constraints, some payloads are further limited by some of their subsystems, such as power amplifiers. In this work we assume the satellite is equipped with N_a power amplifiers, with $N_a \leq N_b$. Every amplifier is connected to a certain number of beams, but no beam is connected to more than one amplifier. We consider these connections can not be changed during operation. The amplifiers also impose a maximum power constraint: the sum of the power allocated to a group of beams connected to amplifier a can not exceed a certain amount P_a^{max}.

Taking all the constraints into account, the problem is formulated as follows

$$\min_{P_{b,t}} \quad \sum_{t=1}^{T} f(\mathcal{P}_t, \mathcal{D}_t) \tag{1}$$

$$\text{s.t.} \quad P_{b,t} \leq P_b^{max}, \quad \forall b \in \mathcal{B}, \forall t \in \{1, \ldots, T\} \tag{2}$$

$$\sum_{b=1}^{N_b} P_{b,t} \leq P_{tot}, \quad \forall t \in \{1, \ldots, T\} \tag{3}$$

$$\sum_{b \in a} P_{b,t} \leq P_a^{max}, \quad \forall a \in \mathcal{A}, \forall t \in \{1, \ldots, T\} \tag{4}$$

$${}_m(R_{b,t}) \geq {}_M, \quad \forall b \in \mathcal{B}, \forall t \in \{1, \ldots, T\} \tag{5}$$

$$P_{b,t} \geq 0, \quad \forall b \in \mathcal{B}, \forall t \in \{1, \ldots, T\}, \tag{6}$$

where $\mathcal{B}$ and $\mathcal{A}$ represent the sets of beams and amplifiers of the satellite, respectively; and $\mathcal{D}_t$, $\mathcal{P}_t$, and $\mathcal{R}_t$ denote the set of throughput demand, power allocated, and data rate attained per beam at timestep t, respectively.

This formulation takes into account the constraints presented so far: on one hand, constraints (2) and (6) represent the upper and lower bounds for the power of each beam in $\mathcal{B}$ at any given timestep, respectively. On the other hand, constraints (3) and (4) express the limitation given by the satellite's and amplifiers' maximum power, respectively. Then, constraint (5) refers to the necessity of achieving a certain link-margin per beam greater than ${}_M$, which depends on the data rate attained, as we will describe in Section 4. Finally, the objective (1) is a function of the requested data rate and the power allocated that reflects the goal of the problem: successfully serving all customers while minimizing the resource consumption. In the following subsection we define the three metrics that will be used as objective functions.

Objective metrics

Multiple metrics have been used in different works on power allocation algorithms for multibeam HTS. In this study we consider three different objective metrics, all of them based on the power allocation at a given timestep t. We first introduce the total *Aggregated Power* (AP) as a measure of resource consumption. This metric simply adds the power allocated to every beam, specifically,

$$AP_t = \sum_{b=1}^{N_b} P_{b,t}. \tag{7}$$

Second, we use the *Unmet Demand* (UD), defined as the fraction of the demand that is not served given the power allocation. This metric has already been used in various studies [6][9][15][17] and is formulated as

$$UD_t = \sum_{b=1}^{N_b} \max[D_{b,t} - R_{b,t}(P_{b,t}), 0]. \tag{8}$$

The third metric we consider is the *Satisfaction-Gap Measure* (SGM), which is used as objective function in [7]. The SGM is based on the demand and data rate per beam ($D_{b,t}$ and $R_{b,t}$, respectively) and, following a series of transformations, measures the mismatch with respect to an ideal allocation case, ensuring fairness among beams. The SGM takes values in the interval [0, 1], one indicating the best possible performance. Following our formulation, we minimize the negative SGM.

Table 1. **List of algorithms and their respective optimization metrics.**

Algorithm	Optimization type	Metric(s)
GA	Multi-objective	AP and UD
SA	Single objective	SGM
PSO	Multi-objective	AP and UD
DRL	Single objective	f(AP, UD)

For each algorithm, we use the metric or metrics – as objective functions – that show a better convergence towards an optimal solution. Table 1 summarizes the relationship between the algorithms and the metrics used. While GA and PSO allow for multi-objective implementations, SA and DRL are generally single-objective approaches and therefore only one metric can be used. Specifically, in the case of DRL a linear combination of the AP and UD is considered, as explained in Section 3.

3. ALGORITHMS OVERVIEW

In this section we cover the implementation of the algorithms compared in this work. We first present the three metaheuristic algorithms; then introduce the main features of the DRL architecture, considering two different types of neural networks; and finally we explain the concept behind each of the hybrid algorithms considered, SA-GA and PSO-GA.

Genetic Algorithm (GA)

A GA [18] is a population-based metaheuristic optimization algorithm inspired by the natural evolution of the species. The algorithm operates with a set of solutions to the optimization problem known as *population*. Each single solution is called an *individual*. Iteratively, the procedure combines different solutions (*crossing*) to generate new individuals and then selects the *fittest* ones (in terms of the goodness of the solution with respect to the objective function of the problem) to keep a constant population size. On top of that, some individuals might undergo *mutations*, thus changing the actual solution associated with it. This is done to avoid focusing only on one area of the search space and keep exploring potentially better zones. After a certain number of iterations, when the algorithm execution ends, the fittest individual is chosen as the best solution to the problem.

In our implementation, in the context of a GA execution at timestep t, an individual $x_{k,t}$ is defined as an array of power allocations $\{P^k_{1,t}, P^k_{2,t}, \ldots, P^k_{N_b,t}\}$, for $k \geq 1$. To fulfill constraints (2) and (6) we always keep these values between 0 and P_b^{max} for each beam b. The population is then composed by N_p individuals. When generating new individuals by the procedures described above, these are denoted as $x_{N_p+1,t}, x_{N_p+2,t}$, and so on.

Depending on the case, we choose to initialize the population randomly or use the final population from the previous timestep execution, i.e.

$$\{x_{1,t}, \ldots, x_{N_p,t}\} = \{x_{f_1,t-1}, \ldots, x_{f_{N_p},t-1}\} \tag{9}$$

where $f_1, \ldots, f_{N_p}$ are the indexes of the fittest N_p individuals from the GA execution at timestep $t-1$. Then, the genetic operations are implemented as follows:

Crossing: Individuals from the original population are randomly paired and have a probability p_{cx} of being crossed. If so, for each beam, the power allocations of both individuals are compared and with probability p^i_{cx} the BLX- [19] operator is used.

Selection: Since we consider a multiobjective implementation with AP and UD as objective metrics, we use the well-known NSGA-II selection method [20], which prioritizes non-dominated individuals given both objectives.

Mutation: After the offspring are generated, new individuals might undergo a mutation with probability p_{mut}. If so, for each beam, the power allocation is randomly changed with probability p^i_{mut}.

To accelerate the convergence to a global optimum, in both the crossing and mutation operations we take into account whether each beam is underserving or overserving its customers. If a certain beam is underserving we do not allow allocation changes that reduce the power allocated to that beam and vice versa.

Finally, constraints (3) and (4) are taken care after each iteration of the algorithm. If a specific individual does not meet any of the two constraints, the power allocation levels are proportionally reduced until fulfilment. Table 2 shows the value of parameters of the algorithm used in this work. The implementation is supported by the DEAP library [21].

Table 2. GA Parameters.

Parameter	Symbol	Value
Population size	N_p	200
Individual crossing prob.	p_{cx}	0.75
Individual mutation prob.	p_{mut}	0.15
Gene crossing prob.	p^i_{cx}	0.6
Gene mutation prob.	p^i_{mut}	0.02
BLX-		0.2

Simulated Annealing (SA)

SA [22] is a metaheuristic algorithm inspired by the cooling processes in metallurgy, where a metal object is more malleable the higher its temperature is. This method, which focuses on a single solution to the optimization problem, iteratively *perturbates* the optimization variables one by one observing the changes in the objective function during the course. The process is governed by a parameter known as *temperature*. After changing one of the optimization variables, the algorithm evaluates the objective function and, if the solution improves, the change is kept. However, if the solution is not improved, the algorithm might keep it with a certain probability that is larger the higher the temperature is. The temperature is progressively decreased throughout the iterations – the *annealing* process – thus preventing changes that degrade the solution in later stages of the algorithm. This prevents the algorithm from getting stuck in local optima during early iterations.

In this work we use a parallel SA version inspired by [23] in which the core procedure is based on [24]. Similar to the GA case, at timestep t the single solution x_t is defined as $\{P_{1,t}, P_{2,t}, \ldots, P_{N_b,t}\}$, where $0 \leq P_{b,t} \leq P_b^{max}$ is always fulfilled for all $b \in \{1, \ldots, N_b\}$. At the beginning of the procedure this solution is randomly initialized or defined as

$$x_t = x^f_{t-1}, \tag{10}$$

where x^f_{t-1} is the final solution after the SA execution at timestep $t-1$. Once initialized, N_n SA nodes or processes are launched in parallel, all starting from the initial solution. When all these processes finish, the best N_{best} solutions are extracted and the N_n SA nodes start again using the best solutions as initial conditions instead. This process is carried out a fixed number of times or until convergence. After each iteration, the initial and final temperatures are decreased according to the following rule

$$T_{start} \leftarrow T_{start} - T_{red} \cdot k \tag{11}$$

$$T_{end} \leftarrow T_{end} - T_{red} \cdot k, \tag{12}$$

where T_{red} is an input parameter and k is the iteration number.

Regarding the individual SA processes, these follow the six steps described in [24], using the SGM as objective metric, as in [7]. The cooling rate used in this work is linear, as opposed to the original exponential rate, which for our case unnecessarily increases runtime. Specifically, the cooling process follows

$$T \leftarrow T - T_{step}, \tag{13}$$

where T is the temperature at a given moment of the execution and T_{step} is an input parameter. All the parameters relevant to the algorithm and the corresponding values used are listed in Table 3. For the varying and step vectors, all elements are initialized identically with values v_i and c_i, respectively.

Particle Swarm Optimization (PSO)

The PSO [25] is a metaheuristic iterative algorithm inspired by the search processes insect swarms and bird flocks carry out in the nature. The algorithm focuses on a set of *particles* known as the *swarm*. At every iteration, each of these particles has a *position* and a *velocity*, them being a specific solution to the optimization problem and the direction where the particle is moving to, respectively. Each particle moves between positions – evaluates different solutions – in order to find the best solution given an objective metric. To that end, every particle takes into account its best position found so far (*local best*) as well as the best position found by any member of the swarm (*global best*). At every iteration, the velocity of each

Table 3. SA Parameters.

Parameter	Symbol	Value
Number of nodes [23]	N_n	20
Initial Start temperature	T_{start}	1
Initial Final temperature	T_{end}	$10^{\ 5}$
Temperature step	T_{step}	0.25
Temperature reduction	T_{red}	$100 \cdot T_{step}$
Iterations per variation [24]	N_s	4
Iterations per temp. reduction [24]	N_T	2
Varying vector C [24]	c_i	2
Step vector V [24]	v_i	1
Number of best individuals [23]	N_{best}	5

particle is determined as a function of its local best and the global best. When the iterations end, the global best is selected as the solution to the optimization problem.

In the context of a PSO execution at timestep t, the position of a particle k is denoted by $x_{k;t}$ and defined as an array of power allocations $x_{k;t} = \{P^k_{1;t}; P^k_{2;t}; :::; P^k_{N_b;t}\}$. Each element in the array is kept between 0 and P_b^{max}. Similarly, the velocity is denoted as $v_{k;t}$ and defined as the array $v_{k;t} = \{v^k_{1;t}; v^k_{2;t}; :::; v^k_{N_b;t}\}$. Each value fulfills the following condition

$$-v_b^{max} \leq v^k_{b;t} \leq v_b^{max}; \quad \forall b; k \tag{14}$$

where v_b^{max} is a positive value and the maximum component-wise speed for beam b.

Then, the local and global best vectors are denoted by $x^k_{L;t}$ and $x_{G;t}$, respectively. Note the local best depends on the particle k considered whereas the global best is the same across the whole swarm. Following these definitions, at every iteration the velocity of every particle k is updated as

$$v_{k;t} \leftarrow \ _I v_{k;t} + \ _G \ _G(x_{G;t} - x_{k;t}) + \ _L \ _L(x^k_{L;t} - x_{k;t}) \tag{15}$$

where $_I$ is the inertia factor influence [26], $_G$ is the global factor influence, $_L$ is the local factor influence, $_G$ is the global factor multiplier, and $_L$ is the local factor multiplier. The factor multipliers, which are randomly sampled at every iteration, increase the exploration capabilities of the algorithm, avoiding the whole swarm to be easily stuck in a local optima. Once the velocity is defined, the position is updated as follows

$$x_{k;t} \leftarrow x_{k;t} + v_{k;t} \tag{16}$$

Then, the new position of each particle is evaluated and the local and global bests updated if necessary.

Further considerations of the algorithm include common elements with the GA implementation, such as the use of underserving and overserving heuristics and additional mutation operations to increase exploration. Then, constraints (3) and (4) are also taken care of after every iteration and the multi-objective implementation is based on the work in [27]. Finally, the DEAP library [21] is also used as a support. The full list of parameters considered for this algorithm is shown in Table 4[2].

[2]**The intertia factor influence is set to the optimal value found in [26]**

Table 4. PSO Parameters.

Parameter	Symbol	Value
Number of particles	N_p	500
Inertia factor influence [26]	$_I$	0.729844
Local factor influence	$_L$	2
Local factor multiplier	$_L$	$U(0;1)$
Global factor influence	$_G$	2
Global factor multiplier	$_G$	$U(0;1)$
Max. component-wise speed	v_b^{max}	$0.025 P_b^{max}$
Particle mutation prob.	p_{mut}	0.1
Component mutation prob.	p^i_{mut}	0.01

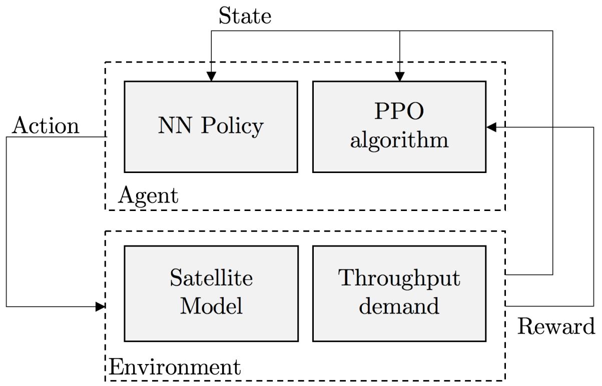

Figure 1. DRL Architecture.

Deep Reinforcement Learning (DRL)

DRL architectures are the enhancement of classic Reinforcement Learning (RL) structures using neural networks. The typical RL setting consists of an agent interacting with an environment [28]. At a certain timestep t, the agent observes the environment's state s_t and takes the action a_t that will maximize the discounted cumulative reward G_t, defined as

$$G_t = \sum_{k=t} {}^{k\ t} r_k \tag{17}$$

where r_k is the reward obtained at timestep k, is the discount factor, and T is the length of the RL episode. An episode is a sequence of states $\{s_0; s_1; :::; s_T\}$ in which the final state s_T is terminal, i.e., no further action can be taken. The goal of the agent is to find a policy that optimally maps states to actions. In the case of DRL, the policy is parametrized and defined by $(a_t|s_t)$, where represents the different weights and biases of the neural network considered.

In this paper we consider the same DRL architecture used in [15], which is depicted in Figure 1. The environment is composed by the satellite model and the throughput demand, whereas the agent is responsible for allocating the power at every timestep and therefore comprises the neural network (NN) policy and the policy optimization algorithm. Having continuous state and action spaces, in this study we use a Policy Gradient [29] method known as the Proximal Policy Optimization (PPO) algorithm [30]. Exploiting the temporal correlation of the DRM problem, we define the *state* of the environment at timestep t as

$$s_t = \{\mathcal{D}_t; \mathcal{D}_{t\ 1}; \mathcal{D}_{t\ 2}; \mathcal{P}_{t\ 1}; \mathcal{P}_{t\ 2}\} \tag{18}$$

where $\mathcal{D}_t$ $\mathcal{D}_{t\quad 1}$, and $\mathcal{D}_{t\quad 2}$ correspond to the set of throughput demand at timesteps t, $t-1$, and $t-2$, respectively; and $\mathcal{P}_{t\quad 1}$ and $\mathcal{P}_{t\quad 2}$ are the optimal power allocations for the two previous timesteps. The initial state is set to $s_0 = \{\mathcal{D}_0; \mathbf{0}; \mathbf{0}; \mathbf{0}; \mathbf{0}\}$.

In this work we assume the agent has access to the optimal power allocation for all the previous timesteps prior to the current allocation decision, which is easy to compute using the inverse equations to (21) - (24), presented in section 4. This way, for every beam b, we can determine the minimum power $P_{b;t}$ that satisfies $R_{b;t} \geq D_{b;t}$. If there is a certain beam b in which the demand can not be met, the optimal power equals its maximum allowed power P_b^{max}.

The *action* of the agent is simply allocating the power for each beam, therefore

$$a_t = \{P_{b;t} | b \in \{1; :::; N_b\}; \; 0 \leq P_{b;t} \leq P_b^{max}\}: \qquad (19)$$

As shown in Table 1, in the DRL case we consider a function of the AP and UD metrics. This is reflected in the *reward* of the architecture, defined as

$$r_t = -\frac{\sum_{b=1}^{N_b} \max(D_{b;t} - R_{b;t}; 0)}{\sum_{b=1}^{N_b} D_{b;t}} - \frac{\sum_{b=1}^{N_b} (P_{b;t} - P_{b;t})^2}{\sum_{b=1}^{N_b} P_{b;t}} \qquad (20)$$

where   is a weighting factor, $P_{b;t}$ is the power set by the agent, $P_{b;t}$ is the optimal power, $R_{b;t}$ is the data rate achieved after taking the action, and $D_{b;t}$ is the demand of beam b at timestep t. Both the data rate and the optimal power are computed using (21) - (24). The reward function plays a relevant role in the training stage of the DRL algorithm; the better it represents the goals of the problem the better the performance in the test stage will be. Therefore, we choose to provide information of the optimal solution in the training stage, which is not available during the test stage. The performance of the DRL approach is exclusively evaluated on the test data.

This definition takes into account both objectives of the problem. The first element focuses on demand satisfaction while the second element responds to the necessity of reducing power. Both elements are normalized by the overall demand and the total optimal power, respectively. The constant   is used to define a priority hierarchy between both objectives. In this work we prioritize smaller values of UD, therefore focusing on the range ≥ 1. According to the reward definition, $r_t \leq 0; \forall t$.

In this work we consider two different neural networks:

Multilayer Perceptron (MLP): Four fully-connected layers with $15N_b$ hidden units each and Rectified Linear Units (ReLU) activations. We also use normalization layers after each hidden layer to reduce training time.

Long Short-Term Memory (LSTM): We use a $5N_b$-dimension array to model the hidden state. Normalization layers are also considered.

Finally, for all DRL simulations we use OpenAI's baselines [31] using the PPO parameters listed in Table 5.

Hybrid algorithms

Finally, given their success in other studies, in this work we also consider two hybrid metaheursitic approaches, namely:

Table 5. **DRL Parameters.**

Parameter	Symbol	Value
Discount factor		0.01
Learning rate	$_r$	0.03
Steps per update	N_{steps}	64
Training minibatches per update	N_{batch}	8
Training epochs per update	N_{epoch}	4
Advantage est. disc. factor [30]		0.8
Clip coef. [30]	$_1$	0.2
Gradient norm clip coef. [30]	$_2$	0.5
Weight factor	(Eq. (20))	100

SA-GA hybrid: This algorithm starts with the SA implementation for a predefined amount of iterations or until convergence. Instead of selecting the best solution from the parallel processes, these are transformed into the initial population of a GA execution. Then, GA carries out another batch of iterations and the final solution is extracted.

PSO-GA hybrid: Since both individual algorithms are based on multi-solution approaches – swarms and populations, respectively – after an initial PSO execution the set of final solutions is transitioned to a GA execution and being used as initial conditions. Then, the final solution is extracted from the final population of GA.

The rationale behind these two hybrids is to complement each other and mitigate the weaknesses of the two individual algorithms. On one hand, SA and PSO are faster algorithms that, even though they implement exploration heuristics, have a high chance of getting stuck in local optima. On the other hand, GA is an algorithm that, despite being slower, if correctly implemented guarantees asymptotic optimality. The objective of the hybrid methods is to first, obtain a "fast and good" solution, and then slowly reiterate on that solution to approach it to the global optimum of the problem.

For both approaches we use the parameters already presented for each of the individual algorithms. For all the simulations that follow in this work, we consider 3 initial iterations of the SA followed by the GA execution in the case of the SA-GA hybrid, and 50 initial iterations of the PSO for the PSO-GA hybrid.

4. Environment Models

This section presents the different models that are used in the simulations in Section 5. First, we introduce the satellite model; next, we cover the link budget model that governs the metrics computation; and finally, we address the different demand models that conform the test scenarios of this study.

Satellite Model

We consider a 200-beam GEO satellite located over America and part of the Atlantic Ocean, with all beams predefined and fixed ($N_b = 200$). Consequently, all beams also have a predefined bandwidth allocation, which we assume uses the Ka band and has been perfectly arranged to minimize interference among beams.

Although multiple users might be served by a single beam, we aggregate all the individual data rate demands and consider there is a single "super user" located precisely at the center

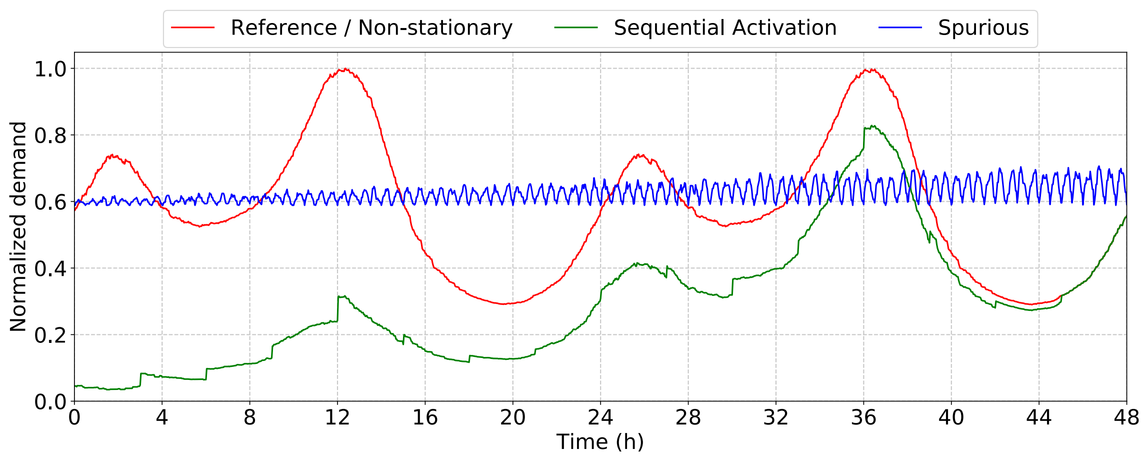

Figure 2. Normalized aggregated demand plot for the four scenarios considered.

of the beam, with a data rate demand equal to the sum of the individual ones.

Link Budget Model

This subsection presents the equations that we use to compute the data rate attained for a specific beam b at timestep t, $R_{b;t}$, given a power allocation $P_{b;t}$. The link budget model used is a parametric model based on [9]. In this work we only show the relevant equations to compute the objective metrics, a deeper description of the elements present in a satellite communications setting can be found in [32].

At a given timestep t, the link's carrier to noise spectral density ratio, $C{=}N_0$, can be computed as

$$\frac{C}{N_0}_{b;t} = P_{b;t} - \text{OBO} + G_{T_x} + G_{R_x} - \text{FPSL}$$
$$- 10\log_{10}(kT_{sys}); \qquad \text{[dB]} \qquad (21)$$

where $P_{b;t}$ is the power (dB) allocated to beam b at timestep t, OBO is the power-amplifier output back-off (dB), G_{T_x} and G_{R_x} are the transmitting and receiving antenna gains (dB), respectively, FSPL is the free-space path loss (dB), k is the Boltzmann constant, and T_{sys} is the system temperature (K). We assume the antenna and amplifier architecture is constant and therefore the antenna gains and OBO are constant through the simulation.

Then, the link's $E_b{=}N$ is computed as

$$\frac{E_b}{N}_{b;t} = \frac{C}{N_0}_{b;t} \cdot \frac{BW}{R_{b;t}} \qquad (22)$$

where BW is the bandwidth allocated to beam b (Hz) and $R_{b;t}$ is its data rate achieved at timestep t, which is in turn computed as

$$R_{b;t} = \frac{BW}{1 + {}_r} \cdot \quad \frac{E_b}{N}_{b;t} \; ! \; ; \qquad \text{[bps]} \qquad (23)$$

where ${}_r$ is the roll-off factor and is the spectral efficiency of the modulation and coding scheme (MODCOD) (bps/Hz), which is a function of $E_b{=}N$ itself. We assume Adaptive Coding and Modulation (ACM) mechanisms are considered, and therefore the MODCOD used on each link is the one that

Table 6. Link Budget Parameters.

Parameter	Symbol	Value
TX antenna gain	G_{T_x}	50.2 - 50.9 dB
RX antenna gain	G_{R_x}	39.3 - 40.0 dB
Free-space path losses	FSPL	209.0 - 210.1 dB
Boltzmann constant	k	$1.38 \cdot 10^{\ 23}$ J/K
Roll-off factor	${}_r$	0.1
Link margin	${}_M$	0.5 dB

provides the maximum spectral efficiency while satisfying the following condition

$$\frac{E_b}{N}_{\text{th}} + {}_M \leq \frac{E_b}{N}_{b;t}; \qquad \text{[dB]} \qquad (24)$$

where $\frac{E_b}{N}_{th}$ is the MODCOD threshold (dB), ${}_M$ is the desired link margin (dB), and $\frac{E_b}{N}_{b;t}$ is the actual link energy per bit to noise ratio (dB) computed using equation (22). Equation (24) represents constraint (5) such that

$$_m(R_{b;t}) = \frac{E_b}{N}_{b;t} - \frac{E_b}{N}_{th} \geq {}_M: \qquad (25)$$

Any of equations (24) and (25) validates if a certain power allocation is feasible (i.e., there needs to be at least one MODCOD scheme such that these inequalities are satisfied).

Finally, in this study we use the standards DVB-S2 and DVB-S2X for the MODCOD schemes and therefore the values for and $\frac{E_b}{N}_{th}$ are those tabulated in the DVB-S2X standard definition [33]. The rest of the parameters of the model can be found in Table 6. Some of these parameters have constant values for all beams; others do not and therefore the range for each of them is shown.

Demand Models

In this work we use traffic models that represent realistic operation scenarios. We consider four different datasets that provide measurements for all beams and span through a sufficiently long time window. These were provided by SES S.A. and are composed by N_b time series containing 1,440

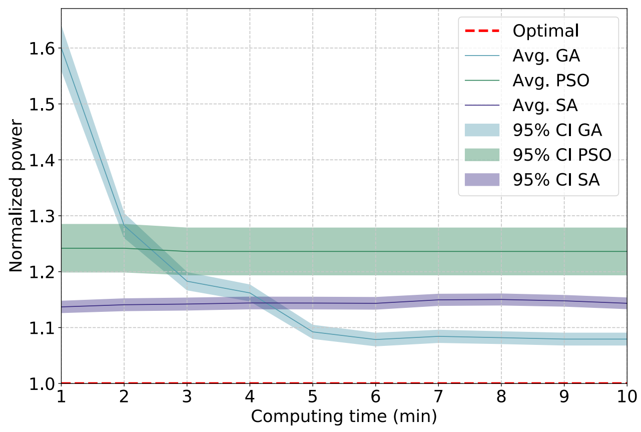

Figure 3. Average aggregated power and 95% CI against computing time allowed. Power is normalized with respect to the optimal aggregated power. Reference scenario used.

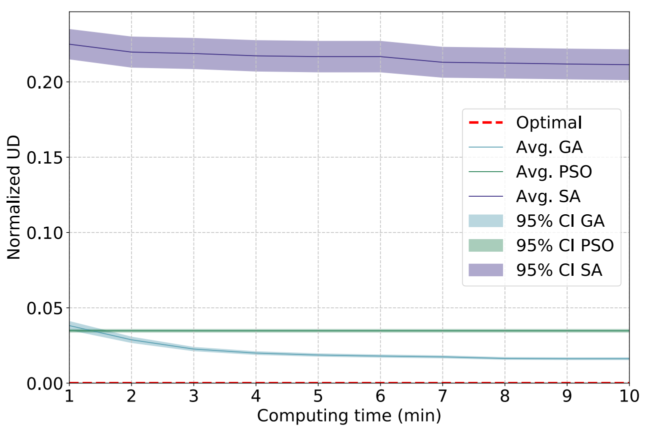

Figure 4. Average aggregated UD and 95% CI against computing time allowed. UD is normalized with respect to the aggregated demand. Reference scenario used.

data points that correspond to demand samples throughout a 48-hour activity period (a sample every 2 minutes). With these datasets, we aim to account for, first, the demand behaviour during a typical daily operations cycle; and second, unfrequent cases that might lead to important service failures if the algorithms do not provide adequate outcomes. The aggregated demand plot for each dataset is depicted in Figure 2 and their respective descriptions are the following:

Reference dataset: represents the throughput demand during a typical operation scenario, in which more data rate is requested during specific time periods in the day.

Sequential activation dataset: considers the scenario in which beams activate progressively throughout a 48-hour time window. As appreciated in Figure 2, the algorithms have to deal with sparse demand at the beginning of the simulation and adapt to the sequential activation of new beams. To create this dataset we take the Reference dataset and create the activation procedure on top of that.

Spurious dataset: accounts for the case in which events that require a relatively high amount of throughput take place in a short amount of time. This dataset is constructed taking a low demand baseline value for all beams and adding random "demand spikes" with a throughput demand between 2 and 5 times the baseline. The frequency of the spikes increases from 2% to 15% with time throughout a 48-hour interval.

Non-stationary dataset: represents a non-stationary scenario in which the tendency of each beam's demand constantly changes. To create this dataset we take the Reference dataset and, in periods of 2 hours, randomly interchange the time series among beams. For instance, two beams b and b^0 swap their demand time series every 2 hours. Consequently, the demand curve looks identical to the Reference one.

5. Results

This section focuses on the different analyses that account for the performance of each algorithm using the scenarios and models presented in the previous section. We address four specific dimensions: time convergence, continuous execution, scalability, and robustness. All simulations run on a server with 20 cores of an Intel 8160 processor, 192 GB of RAM, and one NVIDIA GTX 1080 Ti (used in the DRL training phase). Parallelization has been implemented for all algorithms. For the specific case of DRL, we use 10 environments in parallel to acquire more experience and increase training speed. We compare the results in terms of the power consumed and UD achieved, assigning a higher weight to the latter metric when obtaining a Pareto Front of non-dominated solutions from the population and swarm-based algorithms.

Metaheuristics convergence analyses

The capacity to obtain good solutions in small intervals of time is key to satellite operators, since a higher execution frequency implies a better adaptation to the rapid-changing nature of the demand. In [15] it is proved that DRL, albeit requiring a prior offline training phase, is capable of providing a solution in the range of milliseconds. On the contrary, the three metaheuristic algorithms considered in this work – GA, SA, and PSO – are iteration-based procedures that progressively improve the quality of their solutions [8]. Consequently, the quality of an allocation provided by any of these algorithms is clearly constrained by the computation time available.

In the context of power allocation, the three algorithms have been assessed in terms of optimality in different works [6][7][12]. However, their time-relative performance still needs to be analyzed. To that end, we start by evaluating the convergence rate of the metaheuristic algorithms when applied to the problem considered in this paper. Figure 3 shows the average normalized power (with respect to optimal minimum power) and 95% confidence interval (CI) each of these algorithms allocates depending on the amount of computing time given. Figure 4 shows the UD each algorithm is able to reach (relative to the total demand) under the same premises. Both figures have been obtained after executing the algorithms at 12 different data points uniformly spaced within the first 24h of the Reference dataset. Notice in both figures we denote the optimal value for each metric, which applies to all simulations following.

We can observe two different behaviors: on one hand, both SA and PSO are capable of finding fairly good solutions in less than one minute of computing time but are not able to reach further significant improvements. SA obtains a solution better in power whereas PSO reaches a lower level of UD. On the other hand, GA shows a continuous improvement that leads it to a dominant position both in terms of power and UD. These behaviours reinforce the motivation in the use of hybrid approaches, as introduced in section 3. Finally, it is interesting

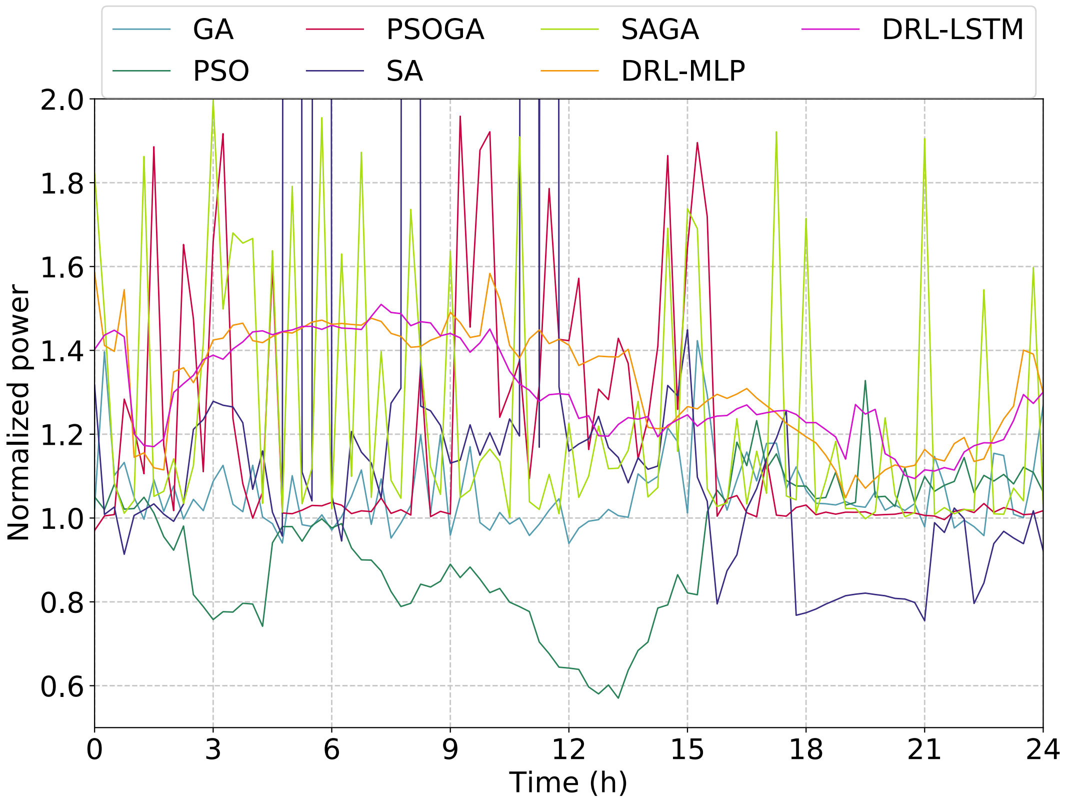

Figure 5. Aggregated power delivered by every algorithm during the continuous execution simulations. Power is normalized with respect to the optimal aggregated power (optimal power is 1). Reference scenario used.

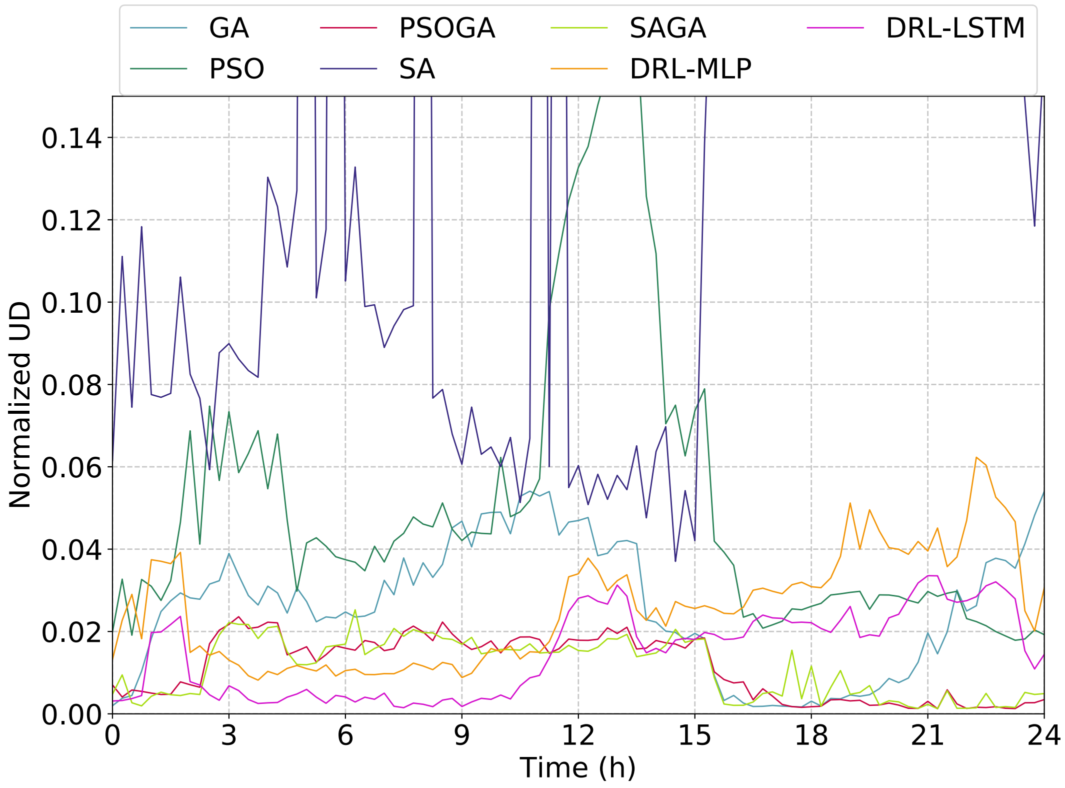

Figure 6. Aggregated UD achieved by every algorithm during the continuous execution simulations. UD is normalized with respect to aggregated demand (optimal UD is 0). Reference scenario used.

to point out that the use of a single-objective optimization metric leads the SA approach to progressively reduce UD and increase power.

These results have a relevant conclusion to bear in mind throughout the remainder of this paper: the performance of the algorithms involving a GA execution is relative to the amount of computing time given. The reader should take into account this limit when extracting conclusions, as allowing for extra computing time could improve the results and vice versa. For the rest of the analyses we will consider a maximum computing time of 3 minutes per algorithm, which is enough for the GA to achieve less than 3% and 20% in UD and allocated to optimal power, respectively.

Continuous execution performance

One of the main gaps found in the literature is the evaluation and comparison of the dynamic algorithms' performance in a scenario that represents a continuous operation context. Prior studies mostly focus on static datasets; only works [14], [15], and [16] prove the usefulness of DRL throughout 24-hour operation scenarios. The desired execution frequency alongside the algorithm convergence time might impose hard constraints on the use of specific candidates, therefore it is important to quantify the algorithm performance in dynamic environments.

To that end, we run each algorithm on the Reference dataset at 97 different timesteps evenly spaced between 0 and 24 hours (15 minutes between timesteps). We impose a 3-minute computing time constraint to all metaheuristic algorithms and, for the DRL simulations, we train each model offline for 6 million timesteps – approximately 7 hours of training time – using the the Reference dataset and the same training and testing procedures described in [15]. Figure 5 shows the aggregated achieved power for every algorithm at each of the timesteps, applying a normalization factor with respect to the optimal aggregated power. Likewise, the UD results for this analysis are shown in figure 6, considering the aggregated demand per timestep as the normalization factor.

We also provide the aggregated performance over time for each algorithm in table 7. Based on these results, we can describe the behaviour of each algorithm as follows:

Table 7. **Aggregated Power and UD results for each algorithm. Power and UD are normalized with respect to the optimal aggregated power and aggregated demand, respectively.**

Algorithm	Power		UD	
	Avg.	Std. dev.	Avg.	Std. dev.
GA	1.06	0.09	0.03	0.02
PSO	0.93	0.16	0.05	0.04
PSO-GA	1.20	0.28	0.01	0.01
SA	2.40	5.77	0.19	0.15
SA-GA	1.25	0.30	0.01	0.01
DRL-MLP	1.32	0.13	0.03	0.01
DRL-LSTM	1.30	0.12	0.01	0.01

GA: depicted in blue, GA shows one of the most stable performances in terms of power, as has one of the smallest power average (6% extra power with respect to the optimal value) and the smallest standard deviation. Regarding UD, despite not being one of the best candidates in terms of this metric, it always serves, on average, at least 97% of the demand.

SA: depicted in purple, it is clearly dominated by other algorithms both in terms of power and UD, achieving the worst results in both metrics. Furthermore, at some timesteps it is not able to find the optimal region and produces allocations that consume more than the double of the power required. This behaviour would not be desirable in a real operation scenario. The main advantage of the SA is its speed to find an initial solution, which clearly has an impact when it is combined with the GA in the SA-GA hybrid (average UD reduction from 19% to 1%).

PSO: depicted in dark green, it shows a similar behaviour to the SA in terms of local-optimality and speed but clearly dominates the annealing method, since, specially during the first 12 hours, achieves a lower UD consuming less power with respect to the SA. Still, the variability of this algorithm produces non-desirable outcomes at several intervals of the simulation that incur in more than 10% of the demand not met (e.g. time interval from 9 to 15 hours).

SA-GA: depicted in light green, the first of the two hybrids shows the positive impact in the UD with respect to any of the individual algorithms alone, as shows an average service rate of 99%. However, in terms of power it preserves part of the variability induced by the SA stage. This originates a ”spiky” behaviour throughout the 24 hours. This amount of variability might not be desirable.

PSO-GA: depicted in red, this algorithm achieves similar low UD results than other algorithms and, as opposed to SA, does not show an excess power use. The main disadvantage is, although in smaller quantity with respect to the SA-GA hybrid, the presence of power spikes.

DRL: depicted in orange (MLP) and pink (LSTM), this approach shows good UD results, serving, on average, 97% (MLP) and 99% (LSTM) of the demand for all timesteps. However, both networks are too conservative in terms of power; although the behaviour is stable and does not show any spike. The greatest advantage of DRL is its capacity to achieve these results in milliseconds, an important feature for rapidly-changing scenarios. Finally, it is worth to point out that, although it is greatly popular among time series prediction applications, the LSTM does not necessarily eclipse the MLP if this latter network is given a good representation for the environment state.

Based on the presented results, we can observe some algorithms are dominated and should not be considered for online real settings. First, both hybrids dominate their individual counterparts in terms of UD (SA-GA dominates SA and PSO-GA dominates PSO, respectively). Between these two, PSO-GA shows less variability in its power allocations. Then, GA is one of the most stable algorithms and its asymptotic optimal behaviour makes it outperform the PSO-GA in terms of power. Finally, in addition to showing remarkable UD performance, both DRL approaches can not be matched in terms of speed, making it the dominant algorithm in that dimension.

Scalability analyses

As pointed out in the introductory section, there is an order of magnitude of difference in the number of beams between the models used in previous power allocation studies and the satellite systems that are expected to be launched in the coming years. While all systems will be able to operate hundreds of beams, some constellations will scale up to thousands of configurable beams. In this part of our analysis we focus on the scaling capabilities of the metaheuristic algorithms, given their computing time constraints. Since DRL has proved its capacity to deal with high-dimensional inputs in other domains

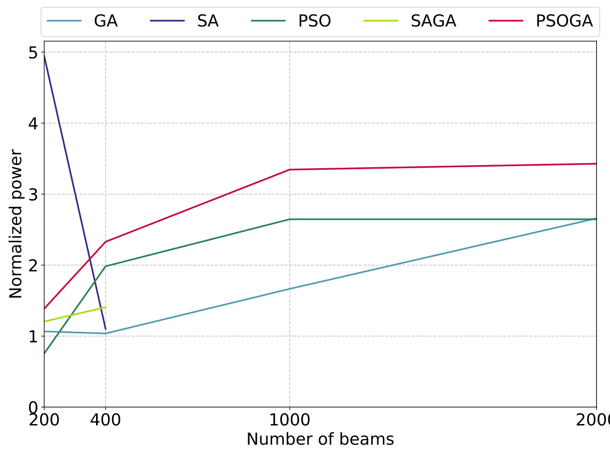

Figure 7. **Average aggregated power against number of beams. Power is normalized with respect to the optimal aggregated power. Reference scenario used.**

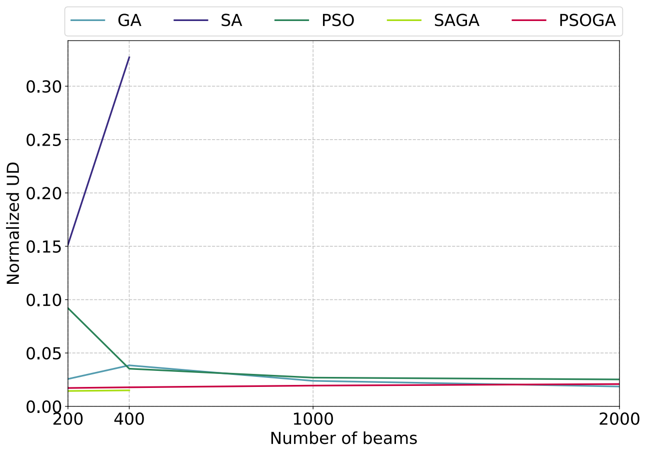

Figure 8. **Average aggregated UD against number of beams. UD is normalized with respect to the aggregated demand. Reference scenario used.**

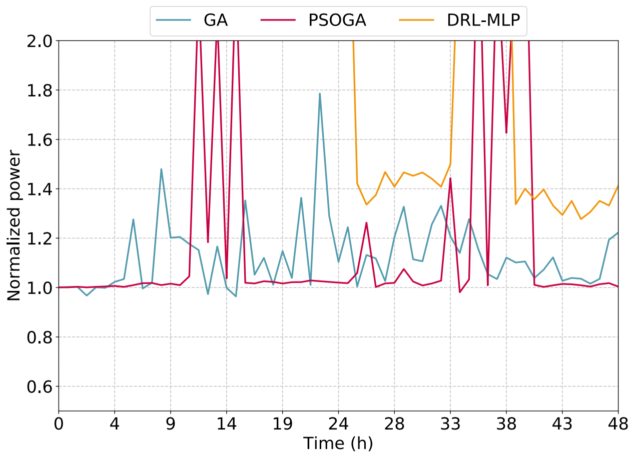

Figure 9. **Aggregated power delivered in a continuous execution using the Sequential activation dataset. Power is normalized with respect to the optimal aggregated power.**

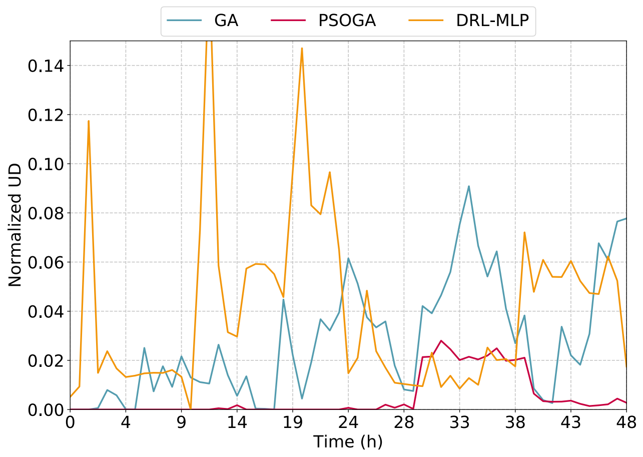

Figure 10. **Aggregated UD achieved in a continuous execution using the Sequential activation dataset. UD is normalized with respect to the aggregated demand.**

[34][35] and we do not address the offline training details in this work, we do not consider it in this part of the paper.

To assess how well each algorithm scales we test the implementations on satellites with an increasing number of beams. To that end, we take our 200-beam model described in section 4 and replicate it, including beams, amplifiers, and users. When replicating the model once, a new 400-beam satellite is created and it can be controlled with any of the algorithms considered in this study. In this subsection we take this replicating action several times and create three new models with 400, 1000, and 2000 beams, respectively.

Figure 7 shows the average aggregated power obtained per algorithm in each of the scaled models. To obtain the curves we use the Reference dataset and run each algorithm at 25 different timesteps evenly spaced between the 10th and 16th hour (15 minutes between timesteps). We choose this time window since it corresponds to the demand peak. Again, we limit the computing time to 3 minutes. Likewise, Figure 8 shows the average aggregated UD under the same premises.

The results show that not all the algorithms can be scaled given the time restrictions imposed. That is the case of SA and the hybrid SA-GA. Since, as opposed to PSO and GA, SA is a sequential algorithm – it only alters one variable between consecutive evaluations of the objective function – its performance is clearly limited by the number of variables in the model. Consequently, for the 1000-beam and 2000-beam cases the algorithm is not able to finish the first iteration on time.

On the contrary, GA, PSO, and their hybrid do obtain solutions for the scaled models. We can observe that, while maintaining a UD result below 5%, these algorithms incur in greater power excesses when increasing the number of controlled beams. Having more variables to tune, and therefore bigger search and exploration spaces, makes these algorithms carry out less efficient iterations, which has an impact on the power required to reach minimum UD solutions.

At this point we have a good overview of the strengths and weaknesses of each algorithm when applied to the DRM problem. Among the six different methods, we identify three that outperform the rest in one dimension involved in this problem. First, we acknowledge the ability of the hybrid PSO-

Table 8. Aggregated Power and UD results for each algorithm and robustness analysis. Power and UD are normalized with respect to the optimal aggregated power and aggregated demand, respectively.

Algorithm	Power		UD	
	Avg.	Std. dev.	Avg.	Std. dev.
Sequential Activation				
GA	1.13	0.14	0.03	0.02
PSO-GA	1.20	0.44	0.01	0.01
DRL-MLP	2.51	1.23	0.04	0.04
Spurious				
GA	0.95	0.14	0.02	0.01
PSO-GA	1.99	0.54	0.01	0.00
DRL-MLP	0.99	0.12	0.12	0.02
Non-stationary				
GA	1.01	0.17	0.06	0.03
PSO-GA	1.37	0.57	0.02	0.01
DRL-MLP	1.16	0.37	0.12	0.10

GA to obtain the smallest values of UD under the equal time constraints. Second, the solo GA algorithm shows one of the most stable behaviors, which leads it to obtain the best power-UD ratio in most cases. Finally, the speed of DRL can not be matched by any of the other algorithms, both networks prove its ability to obtain a great UD performance in less than one second. Consequently, in the following section we only focus on these three approaches.

Robustness analyses

One of the most important features a DRM algorithm must account for is robustness. While it is crucial that the algorithms perform at their best during the typical daily operation, being able to tackle the not-so-frequent events is also relevant. In this subsection we look away from the Reference dataset and focus on the performance of the algorithms on the sporadic use cases represented by the three other datasets presented in the previous section. Given their dominance in the previous analyses, in this part of our study we only consider GA, DRL, and the hybrid PSO-GA. Regarding DRL, we only consider the MLP network, since both the MLP and the LSTM have showed closely similar behaviour. For all datasets, we take 61 timesteps evenly spaced in the 48-hour time window and limit the computing time to 3 minutes. We now discuss each simulation in detail; the aggregated results for the three algorithms and test cases are shown in Table 8.

We start considering the *Sequential activation* dataset; figures 9 and 10 show the normalzied power and UD results for this scenario, respectively. We can observe that while PSO-GA dominates the UD results, GA presents a better, more robust power performance. Regarding the latter algorithm, when taking the previous timestep solution as an initial condition for the following timestep, it "assumes" the same amount of beams is active, which when that is not true leads to unnecessary increases in UD, as it does not have enough time to find the new optimal area in the search space (e.g. spike between 19th and 24th hour). Since its training process involved the Reference dataset – which has a higher average demand per beam –, we note that DRL is biased towards allocating high amounts of power, specially during the first 24 hours of the use case, when less than half of the beams are active.

We continue by addressing the *Spurious* dataset. The power and UD performance of the algorithms in this scenario is given by figures 11 and 12, respectively. In this case we can appreciate a non-dominance situation between PSO and GA, the former achieves the best UD result and the latter reaches way lower levels of consumed power. PSO-GA shows the worse levels of power standard deviation in this case. As opposed to the Sequential activation scenario, DRL does not allocate high amounts of power this time, although attains a worse UD performance. Since this is a mostly "flat" input that strongly deviates from the training data, the neural network performs poorly. This result highlights the importance of a varied offline training stage for the DRL approach.

Finally, we focus on the *Non-stationary* dataset. While this is a highly unlikely dataset, it is important to quantify the performance of the algorithms in cases in which the intrinsic behaviour of the demand changes. Figures 13 and 14 show the power and UD performance for the three algorithms considered, respectively. We can clearly appreciate that non-stationarity affects and degrades the performance of the three algorithms, specially in standard deviation, with respect to the continuous execution analyses. First, GA achieves the smallest standard deviation when it comes to power consumption but incurs in higher UD penalties compared to its performance in the Reference dataset (6% vs. 3%). Second, the PSO-GA hybrid still achieves the best UD result but shows the worst power variability and a considerable amount of power spikes that correspond to more than twice the optimal power in some cases. Finally, given the demand magnitude similarity between the Reference and Non-stationary datasets, DRL does not show any power bias but its performance is clearly worsened with respect to the previous results, specially when it comes to UD (12% vs. 3%). DRL is only able to obtain good UD results at the timesteps where the assignation of each demand time series to a beam matches the one used in the training dataset.

These results emphasize the need to evaluate the performance of the algorithms from different perspectives. Approaches that show a strong UD performance such as DRL and the PSO-GA hybrid turn out to be less robust when the demand behaviour differs from the training data in the case of DRL or shows a bigger relative change than PSO-GA can afford. The response of both algorithms to these "new" scenarios is either to allocate excessive amounts of power or incur in higher UD penalties. On the contrary, albeit not being the best option when tested under the Reference dataset premises, GA shows the most robust behaviour, always attaining the least power variability and thus reducing the uncertainty regarding its performance.

6. CONCLUSIONS

In this paper we have presented an in-depth comparison between the most studied Artificial Intelligence-based algorithms for the Dynamic Resource Management problem in flexible high throughput satellites. To that end, we have focused on the dynamic power allocation problem and considered the following algorithms: Genetic Algorithms (GA), Simulated Annealing (SA), Particle Swarm Optimization (PSO), Deep Reinforcement Learning (DRL), and the hybrid approaches combining the annealing and swarm methods with the GA implementation, respectively. We have used a comprehensive link budget model, a 200-beam satellite model, and data provided by a satellite operator. Accounting for the dimensionality and complexity of the expected satellite communications landscape, we have focused on four characteristics of each method: time convergence, continuous execution, scalability, and robustness. To analyze the results we have considered the

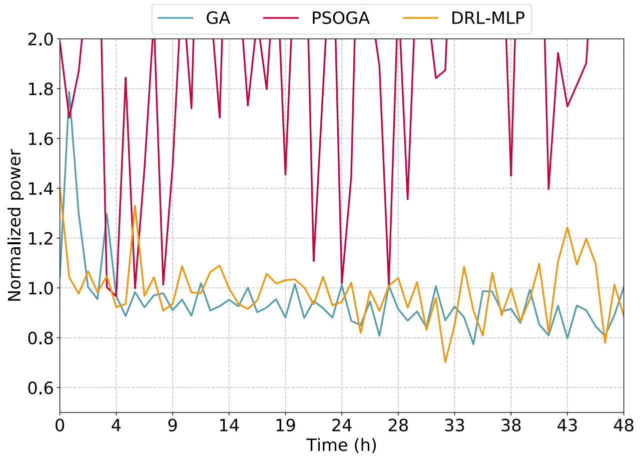

Figure 11. **Aggregated power delivered in a continuous execution using the Spurious dataset. Power is normalized with respect to the optimal aggregated power.**

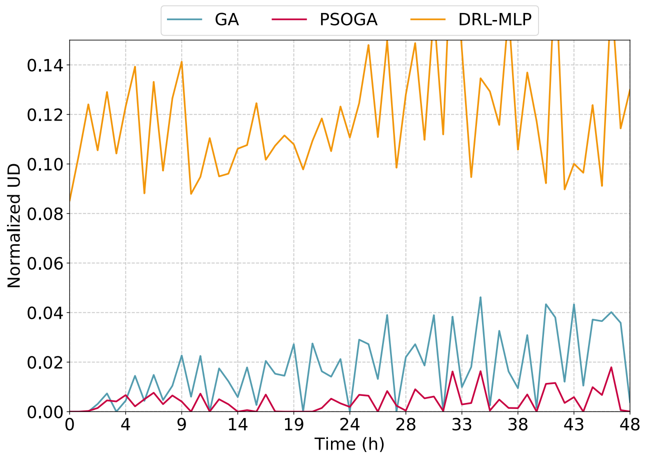

Figure 12. **Aggregated UD achieved in a continuous execution using the Spurious dataset. UD is normalized with respect to the aggregated demand.**

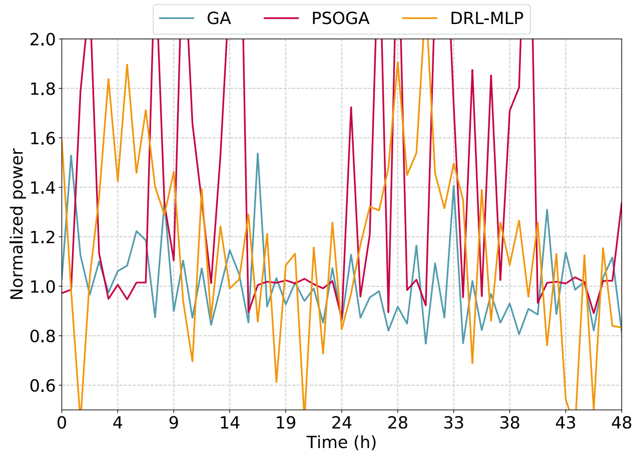

Figure 13. **Aggregated power delivered in a continuous execution using the Non-stationary dataset. Power is normalized with respect to the optimal aggregated power.**

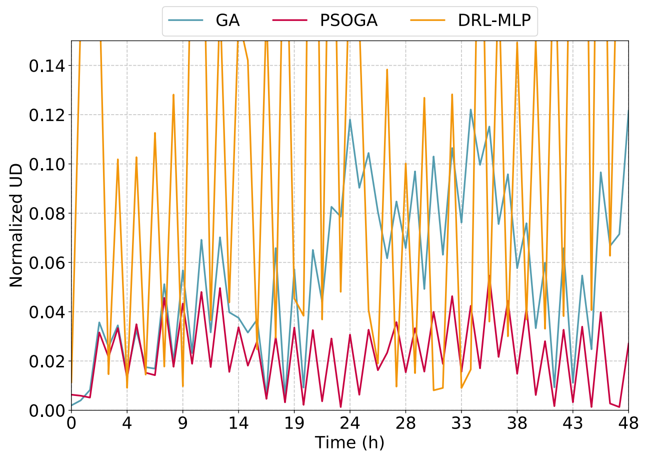

Figure 14. **Aggregated UD achieved in a continuous execution using the Non-stationary dataset. UD is normalized with respect to the aggregated demand.**

total power consumed and the Unmet Demand (UD) as key performance indicators.

The results have showed dominant algorithms among the set. While PSO and SA are fast algorithms in terms of finding a good solution, both fail to find the global optimum and get stuck in local optima. Furthermore, given the sequential behaviour of the latter, its use has been shown to be clearly contrained by the dimensionality of the model used. This issue also affects the performance of the hybrid SA-GA, which although having demonstrated one of the best behaviours in the 200-beam continuous execution analyses, its poor scaling properties make it unfit to be used in systems with large amounts of configurable variables. On the contrary, the hybrid PSO-GA has showed one of the best performances in terms of UD. However, its "bursty" behaviour in terms of power needs to be further addressed. GA has proved to successfully handle this issue, since it is the most robust algorithm in terms of power, with a strong performance when it comes to that metric. Finally, DRL, regardless of the neural network used in the architecture, has proved to be the fastest algorithm and when the training process is carried out appropiately its performance in terms of UD matches the hybrids'. However, we have seen robustness is a critical issue for this algorithm, which motivates the further study of the relationship between the training and operation phases.

In light of these results, we identify three potential candidates that can be used during real satellite operations. In cases in which time is critical and there is a strong need to obtain a solution in seconds, implementing a DRL architecture is definitely the best option, specially as it has shown a great service performance on the typical demand behaviour dataset. If time is not an issue, a better option might be using the PSO-GA hybrid, which, on top of a strong UD performance, has shown better – lower – power results. Finally, when the behaviour of the users changes often and therefore the operator can not guarantee a low uncertainty threshold, the robustness of GA makes it the strongest candidate, as it has shown the lowest standard deviation values across all scenarios.

ACKNOWLEDGMENTS

This work was supported by SES. The authors want to thank SES for their input to this paper and their financial support.

The authors would also like to thank Ruben Alinque for his help in the data treatment and William Torous for his inputs in the Simulated Annealing implementation.

References

[1] M. Coleman, “Is AI key to the survival of satcoms?” 2019. [Online]. Available: https://spacenews.com/op-ed-is-ai-key-to-the-survival-of-satcoms/

[2] P. Angeletti, R. De Gaudenzi, and M. Lisi, “From” Bent pipes” to” software defined payloads”: evolution and trends of satellite communications systems,” in *26th International Communications Satellite Systems Conference (ICSSC)*, 2008.

[3] Northern Sky Research, “VSAT and Broadband Satellite Markets,” Tech. Rep., 2019.

[4] M. Guerster, J. J. Garau Luis, E. F. Crawley, and B. G. Cameron, “Problem representation of dynamic resource allocation for flexible high throughput satellites,” in *2019 IEEE Aerospace Conference*, 2019.

[5] N. Abbas, Y. Nasser, and K. E. Ahmad, “Recent advances on artificial intelligence and learning techniques in cognitive radio networks,” *Eurasip Journal on Wireless Communications and Networking*, vol. 2015, no. 1, 2015.

[6] A. I. Aravanis, B. Shankar M. R., P.-D. Arapoglou, G. Danoy, P. G. Cottis, and B. Ottersten, “Power Allocation in Multibeam Satellite Systems: A Two-Stage Multi-Objective Optimization,” *IEEE Transactions on Wireless Communications*, vol. 14, no. 6, pp. 3171–3182, jun 2015.

[7] G. Cocco, T. De Cola, M. Angelone, Z. Katona, and S. Erl, “Radio Resource Management Optimization of Flexible Satellite Payloads for DVB-S2 Systems,” *IEEE Transactions on Broadcasting*, vol. 64, no. 2, pp. 266–280, 2018.

[8] E.-G. Talbi, *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009, vol. 74.

[9] A. Paris, I. del Portillo, B. G. Cameron, and E. F. Crawley, “A Genetic Algorithm for Joint Power and Bandwidth Allocation in Multibeam Satellite Systems,” in *2019 IEEE Aerospace Conference*. IEEE, 2019.

[10] J. Lei and M. A. Vazquez-Castro, “Joint power and carrier allocation for the multibeam satellite downlink with individual SINR constraints,” in *2010 IEEE International Conference on Communications*. IEEE, 2010, pp. 1–5.

[11] H. Wang, A. Liu, X. Pan, and J. Yang, “Optimization of power allocation for multiusers in multi-spot-beam satellite communication systems,” *Mathematical Problems in engineering*, vol. 2014, 2014.

[12] F. R. Durand and T. Abrão, “Power allocation in multibeam satellites based on particle swarm optimization,” *AEU - International Journal of Electronics and Communications*, vol. 78, pp. 124–133, 2017.

[13] P. V. R. Ferreira, R. Paffenroth, A. M. Wyglinski, T. M. Hackett, S. G. Bilen, R. C. Reinhart, and D. J. Mortensen, “Multiobjective Reinforcement Learning for Cognitive Satellite Communications Using Deep Neural Network Ensembles,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 5, pp. 1030–1041, 2018.

[14] X. Hu, S. Liu, R. Chen, W. Wang, and C. Wang, “A Deep Reinforcement Learning-Based Framework for Dynamic Resource Allocation in Multibeam Satellite Systems,” *IEEE Communications Letters*, vol. 22, no. 8, pp. 1612–1615, 2018.

[15] J. J. Garau Luis, M. Guerster, I. del Portillo, E. Crawley, and B. Cameron, “Deep Reinforcement Learning Architecture for Continuous Power Allocation in High Throughput Satellites,” *arXiv preprint arXiv:1906.00571*, 2019.

[16] X. Hu, S. Liu, Y. Wang, L. Xu, Y. Zhang, C. Wang, and W. Wang, “Deep reinforcement learning-based beam Hopping algorithm in multibeam satellite systems,” *IET Communications*, 2019.

[17] R. Rinaldo, X. Maufroid, and R. C. Garcia, “Non-uniform bandwidth and power allocation in multi-beam broadband satellite systems,” *Proceedings of the AIAA*, 2005.

[18] J. H. Holland and Others, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[19] L. J. Eshelman and J. D. Schaffer, “Real-coded genetic algorithms and interval-schemata,” in *Foundations of genetic algorithms*. Elsevier, 1993, vol. 2, pp. 187–202.

[20] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[21] F. A. Fortin, D. Rainville, M. A. G. Gardner, M. Parizeau, and C. Gagné, “DEAP: Evolutionary Algorithms Made Easy,” *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, 2012.

[22] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing,” *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[23] D. J. Ram, T. H. Sreenivas, and K. G. Subramaniam, “Parallel simulated annealing algorithms,” *Journal of parallel and distributed computing*, vol. 37, no. 2, pp. 207–212, 1996.

[24] A. Corana, , and M. C. Martini and S. Ridella, “Minimizing Multimodal Functions of Continuous Variables with the ”Simulated Annealing” Algorithm,” *ACM Transactions on Mathematical Software*, vol. 15, no. 3, p. 287, 1989.

[25] R. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” in *MHS’95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. Ieee, 1995, pp. 39–43.

[26] Y. Shi and R. Eberhart, “A modified particle swarm optimizer,” in *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*. IEEE, 1998, pp. 69–73.

[27] C. A. C. Coello and M. S. Lechuga, “MOPSO: A proposal for multiple objective particle swarm optimization,” in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC’02 (Cat. No. 02TH8600)*, vol. 2. IEEE, 2002, pp. 1051–1056.

[28] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[29] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in *Advances in*